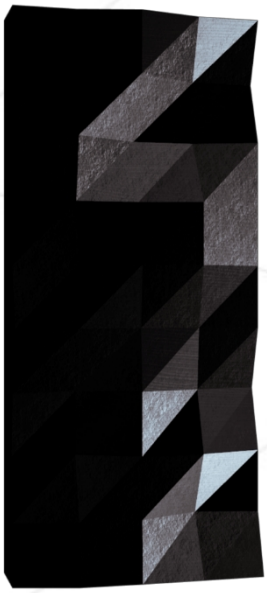

C++ - Modül 05

Tekrar ve İstisnalar

Özet:
Bu doküman C++ modüllerinden Modül 05'in alıştırmalarını içermektedir.

Sürüm: 9.1

İçindekiler

Bölüm I Giriş

C++, Bjarne Stroustrup tarafından C programlama dilinin ya da "C with Classes" (Sınıflı C) dilinin bir uzantısı olarak yaratılmış genel amaçlı bir programlama dilidir (kaynak: Wikipedia).

Bu modüllerin amacı sizi **Nesne Yönelimli Programlama ile** tanıştırmaktır. Bu, C++ yolculuğunuzun başlangıç noktası olacaktır. OOP öğrenmek için birçok dil önerilmektedir. Eski dostunuz C'den türetildiği için C++'ı seçmeye karar verdik. Bu karmaşık bir dil olduğundan ve işleri basit tutmak için kodunuz C++98 standardına uygun olacaktır.

Modern C++'ın birçok açıdan çok farklı olduğunun farkındayız. Dolayısıyla, yetkin bir C++ geliştiricisi olmak istiyorsanız, 42 Common Core'dan sonra daha ileri gitmek size kalmış!

Bölüm II Genel kurallar

Derleme

- Kodunuzu c++ ve -Wall -Wextra -Werror bayrakları ile derleyin
- Eğer -std=c++98 bayrağını eklerseniz kodunuz yine de derlenmelidir

Biçimlendirme ve adlandırma kuralları

- Alıştırma dizinleri şu şekilde adlandırılacaktır: ex00, ex01, ... , exn
- Dosyalarınızı, sınıflarınızı, işlevlerinizi, üye işlevlerinizi ve niteliklerinizi yönergelerde belirtildiği şekilde adlandırın.
- Sınıf adlarını **UpperCamelCase** biçiminde yazın. Sınıf kodu içeren dosyalar her zaman sınıf adına göre adlandırılacaktır. Örneğin: ClassName.hpp/ClassName.h, ClassName.cpp veya ClassName.tpp. Bu durumda, tuğla duvar anlamına gelen "BrickWall" sınıfının tanımını içeren bir başlık dosyanız varsa, adı BrickWall.hpp olacaktır.
- Aksi belirtilmedikçe, her çıktı mesajı bir yeni satır karakteriyle sonlandırılmalı ve standart çıktıda görüntülenmelidir.
- *Güle güle Norminette!* C++ modüllerinde herhangi bir kodlama stili zorunlu değildir. En sevdiğinizi takip edebilirsiniz. Ancak, akran değerlendiricilerinizin anlayamadığı bir kodun not veremeyecekleri bir kod olduğunu unutmayın. Temiz ve okunabilir bir kod yazmak için elinizden geleni yapın.

İzin Verildi/Yasaklandı

Artık C'de kodlama yapmıyorsunuz. C++ zamanı! Bu nedenle:

- Standart kütüphanedeki neredeyse her şeyi kullanmanıza izin verilir. Bu nedenle, zaten bildiklerinize bağlı kalmak yerine, alışkın olduğunuz C işlevlerinin C++-ish sürümlerini mümkün olduğunca kullanmak akıllıca olacaktır.
- Ancak, başka herhangi bir harici kütüphane kullanamazsınız. Bu, C++11 (ve türetilmiş formlar) ve Boost kütüphanelerinin yasak olduğu anlamına gelir. Aşağıdaki fonksiyonlar da yasaktır: *printf(), *alloc() ve free(). Eğer bunları kullanırsanız, notunuz 0 olacaktır ve hepsi bu kadar.

- Aksi açıkça belirtilmedikçe, using ad alanının <ns_name> ve arkadaş anahtar kelimeleri yasaktır. Aksi takdirde notunuz -42 olacaktır.
- **STL'yi sadece Modül 08 ve 09'da kullanmanıza izin verilmektedir.** Bunun anlamı: o zamana kadar **Kapsayıcı** (vektör/liste/harita/ve benzeri) ve **Algoritma** (<algorithm> başlığını içermesi gereken herhangi bir şey) kullanmayacaksınız. Aksi takdirde notunuz -42 olacaktır.

Birkaç tasarım gereksinimi

- C++'da da bellek sızıntısı meydana gelir. Bellek ayırdığınızda (new anahtar sözcüğü), **bellek sızıntılarından** kaçınmalısınız.
- Modül 02'den Modül 09'a kadar, **aksi açıkça belirtilmediği sürece,** dersleriniz **Ortodoks Kanonik Formunda** tasarlanmalıdır.
- Bir başlık dosyasına konulan herhangi bir işlev uygulaması (işlev şablonları hariç) alıştırma için 0 anlamına gelir.
- Başlıklarınızın her birini diğerlerinden bağımsız olarak kullanabilmelisiniz. Bu nedenle, ihtiyaç duydukları tüm bağımlılıkları içermelidirler. Ancak, **include korumaları** ekleyerek çifte içerme sorunundan kaçınmalısınız. Aksi takdirde notunuz 0 olacaktır.

Beni oku

- Gerekirse bazı ek dosyalar ekleyebilirsiniz (örneğin, kodunuzu bölmek için). Bu ödevler bir program tarafından doğrulanmadığından, zorunlu dosyaları teslim ettiğiniz sürece bunu yapmaktan çekinmeyin.
- Bazen bir alıştırmanın yönergeleri kısa görünebilir ancak örnekler, yönergelerde açıkça yazılmayan gereksinimleri gösterebilir.
- Başlamadan önce her modülü tamamen okuyun! Gerçekten okuyun.
- Odin tarafından, Thor tarafından! Beynini kullan!!!

Çok sayıda sınıf uygulamanız gerekecek. En sevdiğiniz metin düzenleyicinizi komut dosyası haline getiremediğiniz sürece bu sıkıcı görünebilir.

Egzersizleri tamamlamanız için size belirli bir miktar özgürlük verilir. Ancak, zorunlu kurallara uyun ve tembellik etmeyin. Pek çok faydalı bilgiyi kaçırırsınız! Teorik kavramlar hakkında okumaktan çekinmeyin.

Bölüm III

Alıştırma 00: Anne, ben büyüyünce bürokrat olmak istiyorum!

	Egzersiz : 00
Anne, büyüdüğümde bürokrat olmak istiyorum!	
Giriş dizini : *ex00/*	
Teslim edilecek dosyalar : `Makefile, main.cpp, Bureaucrat.{h, hpp}, Bureaucrat.cpp`	
Yasak fonksiyonlar : `Hiçbiri`	

`İstisna sınıflarının Ortodoks Kanonik Formda tasarlanması gerekmediğini lütfen unutmayın. Ancak diğer tüm sınıflar zorunludur.`

Ofisler, koridorlar, formlar ve bekleme kuyruklarından oluşan yapay bir kabus tasarlayalım.
Kulağa eğlenceli geliyor mu? Değil mi? Çok kötü.

İlk olarak, bu devasa bürokratik makinenin en küçük dişlisinden başlayın: **Bürokrat**.

Bir **Bürokratın** sahip olması gereken:

- Sabit bir isim.
- Ve **1** (mümkün olan en yüksek not) ile **150** (mümkün olan en düşük not) arasında değişen bir not.

Geçersiz bir not kullanarak bir Bürokrat örnekleme girişiminde bulunulduğunda bir hata mesajı verilmelidir:
ya bir Bureaucrat::GradeTooHighException ya da bir Bureaucrat::GradeTooLowException.

Bu niteliklerin her ikisi için de getter'lar sağlayacaksınız: getName() ve getGrade(). Bürokratın notunu artırmak veya azaltmak için de iki üye fonksiyon ekleyin. Not aralık dışındaysa, her ikisi de kurucu ile aynı istisnaları fırlatacaktır.

Unutmayın. 1. derece en yüksek ve 150. derece en düşük derece olduğundan, 3. derecenin artırılması bürokrata 2. derece vermelidir.

Atılan istisnalar try ve catch blokları kullanılarak yakalanabilir olmalıdır:

```
dene
{
  /* bürokratlarla bazı şeyler yapmak */
}
catch (std::exception & e)
{
  /* istisnayı ele al */
}
```

Aşağıdaki gibi (köşeli parantezler olmadan) bir şey yazdırmak için ekleme (") operatörünün bir aşırı yükünü uygulayacaksınız:

<isim>, bürokrat derecesi <derece>.

Her zamanki gibi, her şeyin beklendiği gibi çalıştığını kanıtlamak için bazı testler yapın.

Bölüm IV

Alıştırma 01: Toplanın, kurtçuklar!

	Egzersiz : 01
Toplanın, kurtçuklar!	
Giriş dizini : *ex01/*	
Teslim edilecek dosyalar : Önceki alıştırmadaki dosyalar + Form.{h, hpp}, Form.cpp	
Yasak fonksiyonlar : Hiçbiri	

Artık bürokratlarınız olduğuna göre, onlara yapacak bir şeyler verelim. Bir yığın form doldurmaktan daha iyi bir faaliyet olabilir mi?

O zaman bir **Form** sınıfı yapalım. İçinde:

- Sabit bir isim.
- İşaretli olup olmadığını gösteren bir boolean (yapıda değildir).
- İmzalamak için sabit bir not gereklidir.
- Ve bunu uygulamak için sabit bir not gereklidir.

Tüm bu özellikler **özeldir**, korumalı değildir.

Form notları Bürokrat için geçerli olan aynı kuralları izler. Bu nedenle, bir form notu sınırların dışındaysa aşağıdaki istisnalar fırlatılacaktır: Form::GradeTooHighException ve Form::GradeTooLowException.

Daha önce olduğu gibi, tüm nitelikler için getter'lar ve formun tüm bilgilerini yazdıran insertion (") operatörünün bir aşırı yüklemesi yazılır.

Ayrıca Form'a parametre olarak Bürokrat alan bir beSigned() üye fonksiyonu ekleyin. Bürokratın notu yeterince yüksekse (gerekli olandan yüksek veya eşitse) form durumunu imzalı olarak değiştirir. Unutmayın, 1. derece 2. dereceden daha yüksektir.
Not çok düşükse, bir Form::GradeTooLowException atılır.

Son olarak, Bureaucrat'a bir signForm() üye işlevi ekleyin. Form imzalandıysa, şöyle bir şey yazdıracaktır:

<bureaucrat> imzalı <form>

Aksi takdirde, aşağıdaki gibi bir şey yazdıracaktır:

<bürokrat> <form>u imzalayamadı çünkü <sebep>.

Her şeyin beklendiği gibi çalıştığından emin olmak için bazı testler uygulayın ve teslim edin.

Bölüm V

Alıştırma 02: Hayır, 28B formuna ihtiyacınız var, 28C'ye değil...

	Alıştırma : 02
Hayır, 28B formuna ihtiyacınız var, 28C'ye değil...	
Dönüş dizini : *ex02/*	
Teslim edilecek dosyalar : `Makefile, main.cpp, Bureaucrat.[{h, hpp},cpp], Bureaucrat.cpp + AForm.[{h, hpp},cpp], ShrubberyCreationForm.[{h, hpp},cpp], + RobotomyRequestForm.[{h, hpp},cpp], PresidentialPardonForm.[{h, hpp},cpp]`	
Yasak fonksiyonlar : `Hiçbiri`	

Artık temel formlara sahip olduğunuza göre, gerçekten bir şeyler yapan birkaç tane daha yapmanın zamanı geldi.

Her durumda, temel sınıf Form soyut bir sınıf olmalıdır ve bu nedenle AForm olarak yeniden adlandırılmalıdır. Formun niteliklerinin özel kalması gerektiğini ve bunların temel sınıfta olduğunu unutmayın.

Aşağıdaki somut sınıfları ekleyin:

- **ÇalılıkOluşturmaFormu**: Gerekli notlar: sign 145, exec 137
  Çalışma dizininde bir <target>_shrubbery dosyası oluşturun ve içine ASCII ağaçları yazın.
- **RobotomyRequestForm**: Gerekli notlar: sign 72, exec 45
  Bazı delme sesleri çıkarır. Ardından, <hedef>in %50 oranında başarılı bir şekilde robotomize edildiğini bildirir. Aksi takdirde, robotominin başarısız olduğunu bildirir.
- **PresidentialPardonForm**: Gerekli notlar: sign 25, exec 5
  <hedef>in Zaphod Beeblebrox tarafından affedildiğini bildirir.

Hepsi yapıcılarında yalnızca bir parametre alır: formun hedefi. Örneğin, eve çalı dikmek istiyorsanız "ev".

Şimdi execute(Bureaucrat const & executor) const üye işlevini temel forma ekleyin ve somut sınıfların form eylemini yürütmek için bir işlev uygulayın. Formun imzalanıp imzalanmadığını ve formu yürütmeye çalışan bürokratın derecesinin yeterince yüksek olup olmadığını kontrol etmeniz gerekir. Aksi takdirde, uygun bir istisna fırlatın.

Gereksinimleri her somut sınıfta mı yoksa temel sınıfta mı kontrol etmek istediğiniz (daha sonra formu çalıştırmak için başka bir işlev çağırmak) size kalmış. Ancak, bir yol diğerinden daha güzeldir.

Son olarak, executeForm(Form const & form) üye fonksiyonunu Bureau- crat'a ekleyin. Formu çalıştırmayı denemelidir. Eğer başarılı olursa, şöyle bir şey yazdırın:

<bureaucrat> çalıştırıldı <form>

Değilse, açık bir hata mesajı yazdırın.

Her şeyin beklendiği gibi çalıştığından emin olmak için bazı testler uygulayın ve teslim edin.

Bölüm VI

Alıştırma 03: En azından bu kahve yapmaktan iyidir

	Egzersiz : 03
En azından kahve yapmaktan iyidir.	
Dönüş dizini : *ex03/*	
Teslim edilecek dosyalar : `Önceki alıştırmalardan dosyalar + Intern.{h, hpp}, Intern.cpp`	
Yasak fonksiyonlar : `Hiçbiri`	

Form doldurmak yeterince can sıkıcı olduğundan, bürokratlarımızdan tüm gün boyunca bunu yapmalarını istemek acımasızlık olur. Neyse ki stajyerler var. Bu alıştırmada, **Stajyer** sınıfını uygulamanız gerekiyor. Stajyerin adı, notu ve kendine has özellikleri yok. Bürokratların önemsediği tek şey işlerini yapmalarıdır.

Bununla birlikte, stajyerin önemli bir kapasitesi vardır: makeForm() işlevi. İki dizgi alır. Bunlardan ilki bir formun adı, ikincisi ise formun hedefidir. Hedefi ikinci parametreye başlatılacak olan bir **Form nesnesine** (adı parametre olarak geçirilen) bir işaretçi döndürür.

Şuna benzer bir şey yazdıracaktır:

Stajyer <form> oluşturur

Parametre olarak aktarılan form adı mevcut değilse, açık bir hata mesajı yazdırın.

if/elseif/else ormanı kullanmak gibi okunaksız ve çirkin çözümlerden kaçınmalısınız. Bu tür şeyler değerlendirme sürecinde kabul edilmeyecektir. Artık Piscine'de (havuz) değilsiniz. Her zamanki gibi, her şeyin beklendiği gibi çalıştığını test etmelisiniz.

Örneğin, aşağıdaki kod "Ben- der "i hedefleyen bir **RobotomyRequestForm** oluşturur:

```
{
    Intern someRandomIntern;
    Form*   rrf;

    rrf = someRandomIntern.makeForm("robotomi isteği", "Bender");
}
```